BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

REMARKS

In response to the Office Action mailed February 19, 2004, Applicants respectfully request reconsideration. Claims 1-24 are currently pending in this Office Action. Claims 17, 19, 22, and 24 are rejected under 35 U.S.C. §102(b) and claims 1-16, 18, 20, 21, and 23 are rejected under 35 U.S.C. §103(a). To further the prosecution of this application, each of the rejections set forth in the Office Action is addressed herein and the application as presented is believed to be in condition for allowance.

Initially, Applicants note that in paragraph one, the Office Action indicates that formal drawings have not yet been filed in this application. Applicant submits herewith a complete set of formal drawings.

In paragraph two, the Office Action objected to the title of the invention as being non-descriptive. Applicants have amended the title of the invention to overcome this objection. Accordingly, withdrawal of the objection to the title of the invention is respectfully requested.

Rejections Under 35 U.S.C. §102

The Office Action rejected claims 17, 19, 22, and 24 under 35 U.S.C. §102(b) as purportedly being anticipated by Alpert (5,621,886). Applicants respectfully traverse this rejection.

Alpert is directed to a method and apparatus for providing efficient software debugging. Figure 2 of Alpert is flow chart which illustrates a method for debugging. As shown in Figure 2, when an event such as an address breakpoint or a branch breakpoint is recognized, the currently executing process is suspended and the execution environment is stored (Col. 8, lines 16-19). Thus, Alpert teaches that processor execution is suspended based on recognition of event, such as an address breakpoint or branch breakpoint. Alpert does not disclose or suggest suspending processor execution based on a stall signal or a stall attribute associated with debug instructions.

Claim 17

Claim 17 is directed to a computer system for executing instructions in a first, user mode and a second, debug mode. The computer system comprises: a first store for holding user instructions; a second store for holding debug instructions, wherein the debug instructions are

held in the second store in association with debug attributes, wherein said debug attributes include a stall attribute; a fetch unit for selectively fetching instructions from the first or second store depending on the mode of the computer system; a decode unit for decoding said instructions and reading said attributes; and a emulation unit which includes control circuitry which cooperates with the decode unit to selectively set the decode unit into a stall state by issuance of a stall signal; wherein the decode unit includes stall control circuitry which is responsive to reading of a stall attribute or receipt of a stall signal from the emulation unit to place the decode unit into a stall state.

The Office Action acknowledges that Alpert does not explicitly disclose a fetch unit and a decode unit, as recited in claim 17. *See* Office Action, paragraph 5. Applicants agree that these limitations are not explicitly disclosed by Alpert.

Additionally, Alpert fails to teach or suggest, "a second store for holding debug instructions, wherein the debug instructions are held in the second store in association with debug attributes, wherein said debug attributes include a stall attribute; a fetch unit for selectively fetching instructions from the first or second store depending on the mode of the computer system" and "a emulation unit which includes control circuitry which cooperates with the decode unit to selectively set the decode unit into a stall state by issuance of a stall signal; wherein the decode unit includes stall control circuitry which is responsive to reading of a stall attribute or receipt of a stall signal from the emulation unit to place the decode unit into a stall state."

Nowhere does Alpert disclose or suggest a stall attribute associated with a debug instruction and a decode unit that includes stall control circuitry which is responsive to reading of the stall attribute. As discussed above, Alpert teaches that processor execution is suspended upon recognition of an address breakpoint or branch break point. This is different from placing a decode unit into a stall state based upon reading a stall attribute that is associated with a debug instruction.

Thus, claim 17 patentably distinguishes over Alpert. Accordingly, it is respectfully requested that the rejection of claim 17 under 35 U.S.C. §102(b) be withdrawn.

Claims 18-23 depend from claim 17 and are patentable for at least the same reasons. Accordingly, it is respectfully requested that the rejections of claims 18-23 be withdrawn.

Claim 24

Claim 24 is directed to a method of setting a stall state of a computer system which comprises a fetch unit for fetching instructions to be executed and a decode unit for decoding said instructions, wherein the stall state is set selectively at the decode unit by reading stall attributes associated with debug instructions in a debug mode, or by receipt of a stall command responsive to certain conditions when executing user instructions in a user mode.

The Office Action acknowledges that Alpert does not explicitly disclose a fetch unit and a decode unit, as recited in claim 24. *See* Office Action, paragraph 8. Applicants agree that these limitations are not explicitly disclosed by Alpert.

Further, As should be clear from the discussion above, Alpert does not disclose or suggest, "setting a stall state of a computer system which comprises a fetch unit for fetching instructions to be executed and a decode unit for decoding said instructions, wherein the stall state is set selectively at the decode unit by reading stall attributes associated with debug instructions in a debug mode."

Alpert discloses that execution of a processor is suspended based upon recognition of an event, but does not teach or suggest that the stall state of a computer system is set by reading stall attributes associated with debug instructions in a debug mode.

Thus, claim 24 patentably distinguishes over Alpert. Accordingly, it is respectfully requested that the rejection of claim 24 under 35 U.S.C. §102(b) be withdrawn.

Rejections Under 35 U.S.C. §103

The Office Action rejected claims 1, 3-9, 11-16, 18, 20, 21, and 23 as purportedly being obvious over Karp (5,748,936) in view of Alpert. The Office Action further rejected claims 2 and 10 as purportedly being obvious over the combination of Karp and Alpert in view of various other references. Each of these rejections is respectfully traversed, as it would not have been obvious to combine Karp and Alpert. Furthermore, even if one were to combine Karp and Alpert, Applicant's claims patentably distinguish over any such combination.

The Combination of References is Improper

One of ordinary skill in the art would not have been motivated to combine Karp and Alpert, as the two references relate to entirely different technical fields. Specifically, Karp is directed to speculative execution in a processor. Alpert is directed towards the unrelated field of performing debugging during execution of instructions by a processor.

MPEP §2143.01 states that, "Obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found in either explicitly or implicitly in the references themselves or in the knowledge generally available to one of ordinary skill in the art." Here, Alpert does not disclose or suggest that the method for debugging described in the reference is suitable for use in a processor which performs speculative execution (i.e., the processor of Karp). Similarly, Karp does not even mention performing debugging, as Karp is directed to performing speculative execution in a processor. Thus, there is no disclosure or suggestion in Karp that the debugging method of Alpert would be suitable for use in the processor of Karp.

Thus, one of skill in the art would not have been motivated to combine Karp and Alpert. Accordingly, it is respectfully requested that the rejection of Applicants' claims under 35 U.S.C. §103(a) be withdrawn.

Applicants' Claims Patentably Distinguish Over the Combination of Karp and Alpert

Even if one were to combine Karp and Alpert, Applicants' claims patentably distinguish over any such combination.

Karp is directed to a processor used to support speculative execution. Karp discloses that the processor includes a memory for storing predicate values, which indicate whether an operation associated with the values should be executed (Col. 5, line 62 – Col. 6, line 3).

The Office Action concedes that Karp does not teach, "an emulation unit including control circuitry which cooperates with the decode unit to selectively control the decode unit to implement step-by-step execution of an instruction sequence." *See* Office Action, paragraph 12. The Office Action further concedes that Karp does teach, "for each committed instruction, a

divert routine is executed by the computer system and for each non-committed instruction the next instruction in the instruction sequence is executed." *See* Office Action, paragraph 14.

The Office Action asserts that Alpert discloses executing a divert routine for each committed instruction and executing the next instruction in the instruction sequence for each non-committed instruction. Applicants respectfully disagree with this assertion.

Alpert does not disclose executing a divert routine for each committed instruction. Alpert teaches executing a debug handler routing only upon recognition of an event, such as an address breakpoint or branch breakpoint (Col. 2, lines 41-48). Executing a divert routine for each committed instruction is different from executing a debug handler based on recognition of an event.

Furthermore, because the processor disclosed by Alpert is not a predicated processor, Alpert does not teach or suggest non-committed instructions. That is, each instruction in the instruction pipeline will be executed, as there is no guard value to prevent an instruction from being executed. The Office Action asserts that because Alpert discloses that a programmer may be able to selectively disable certain types of events from triggering a debug handler (*see* Alpert, Col. 9, lines 40-49), that Alpert discloses non-committed instructions. *See* Office Action, paragraph 14. Applicants respectfully disagree with this assertion.

Even though Alpert discloses that some events (e.g., certain breakpoints) may be disabled, such that they do not cause the processor to suspend execution and execute a debug handler routine, all instructions executed by the processor of Alpert are committed for execution. That is, Alpert does not disclose or suggest that only some instructions in the instruction sequence are to be executed. Therefore, Alpert does not disclose or suggest non-committed instructions. While some instructions in the instruction sequence may not trigger a debug routine (i.e., due to the programmer disabling certain types of events), these instructions are nevertheless committed for execution.

Claim 1

Claim 1 is directed to a computer system for executing predicated instructions wherein each instruction includes a guard, the value of which determines whether or not that instruction is executed. The computer system comprises: a fetch unit for fetching instructions to be executed;

a decode unit for decoding said instructions; at least one pipelined execution unit for executing decoded instructions and being associated with a guard register file holding values of the guards to allow resolution of the guards to be made to determine whether an instruction is committed; and an emulation unit including control circuitry which cooperates with the decode unit to selectively control the decode unit to implement step-by-step execution of an instruction sequence wherein, for each committed instruction, a divert routine is executed by the computer system and for each non-committed instruction the next instruction in the instruction sequence is executed.

As should be clear from the discussion above, neither Alpert nor Karp, taken alone or in combination, discloses or suggests, "an emulation unit including control circuitry which cooperates with the decode unit to selectively control the decode unit to implement step-by-step execution of an instruction sequence wherein, for each committed instruction, a divert routine is executed by the computer system and for each non-committed instruction the next instruction in the instruction sequence is executed," as recited in claim 1.

Thus, claim 1 patentably distinguishes over Karp and Alpert. Accordingly, it is respectfully requested that the rejection of claim 1 under 35 U.S.C. §103(a) be withdrawn.

Claims 2-12 depend from claim 1 and are patentable for at least the same reasons. Accordingly, it is respectfully requested that the rejection of claims 2 -12 under 35 U.S.C. §103(a) be withdrawn.

Claim 13

Claim 13 is directed to a method for executing predicated instructions wherein each instruction includes a guard, the value of which determines whether or not that instruction is executed. The method comprises: fetching each of a sequence of instructions to be executed; decoding each instruction and requesting resolution of its guard to determine whether the instruction is committed; and if the instruction is committed, implementing a divert routine whereby debug code is executed and, if the instruction is not committed, fetching and decoding the next instruction in the instruction sequence.

As should be clear from the discussion above, neither Alpert nor Karp, taken alone or in combination, discloses or suggests, "if the instruction is committed, implementing a divert

routine whereby debug code is executed and, if the instruction is not committed, fetching and decoding the next instruction in the instruction sequence," as recited in claim 13.

Thus, claim 13 patentably distinguishes over Karp and Alpert. Accordingly, it is respectfully requested that the rejection of claim 13 under 35 U.S.C. §103(a) be withdrawn.

Claims 14-16 depend from claim 13 and are patentable for at least the same reasons. Accordingly, it is respectfully requested that the rejection of claims 14-16 under 35 U.S.C. §103(a) be withdrawn.

CONCLUSION

In view of the foregoing amendments and remarks, this application should now be in condition for allowance. A notice to this effect is respectfully requested. If the Examiner believes, after this amendment, that the application is not in condition for allowance, the Examiner is requested to call the Applicant's attorney at the telephone number listed below.

If this response is not considered timely filed and if a request for an extension of time is otherwise absent, Applicant hereby requests any necessary extension of time. If there is a fee occasioned by this response, including an extension fee, that is not covered by an enclosed check, please charge any deficiency to Deposit Account No. 23/2825.

CERTIFICATE OF MAILING UNDER 37 C.F.R. §1.8(a)

I hereby certify that this document is being placed in the United States mail with first-class postage attached, addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on June 21, 2004.

Gail Donoghue

Attorney Docket No.: S1022.80582US00
X06/19/04
(June 19, 2004 being a Saturday)

Respectfully submitted,

Cofler et al., Applicant

By James

James H. Morris
Reg. No.: 34,681
WOLF, GREENFIELD & SACKS, P.C.
600 Atlantic Avenue
Boston, Massachusetts 02210
Tel. (617) 720-3500